

Das XML Informationset

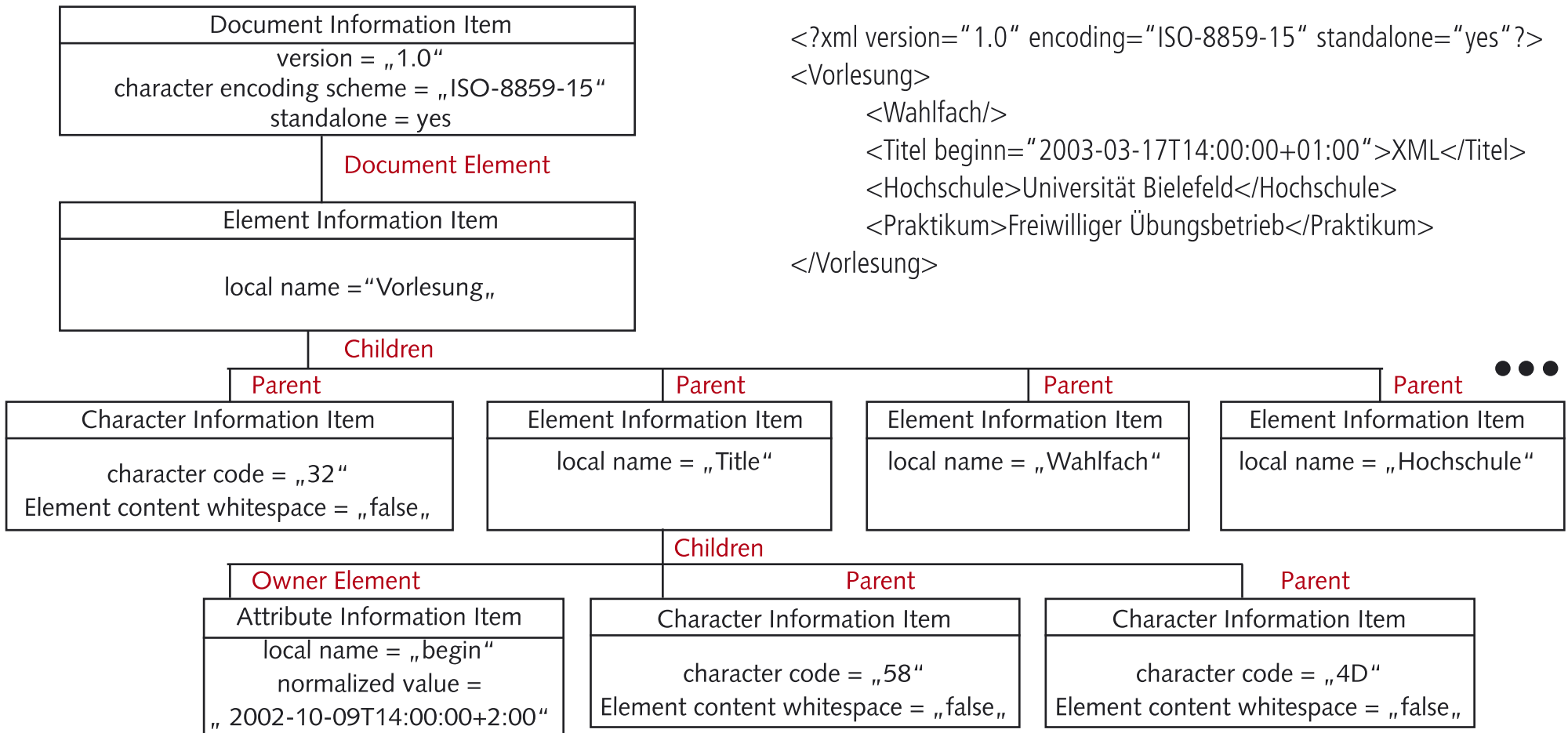
Document Object Model

XOM

Zusammenfassung

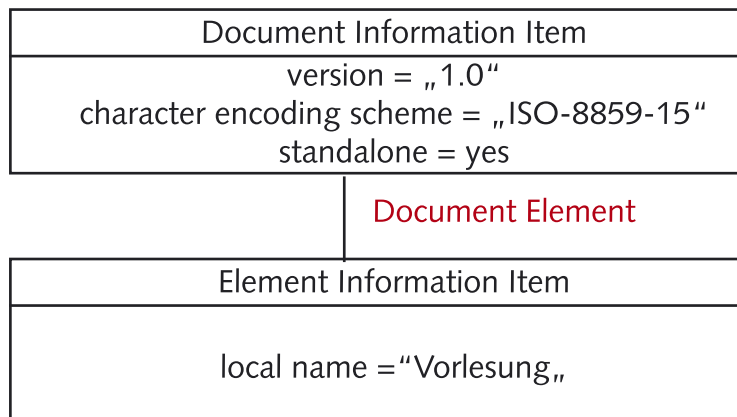
## XML Infoset - Das Strukturelle Grundkonzept

Das XML Infoset definiert die **Semantik**, führt die zentralen **Begriffe** ein, setzt diese in **Beziehung** und definiert so die syntaxunabhängige **Struktur** eines XML Dokuments.



## Document Information Item

- stellt den äußeren Rahmen
- Version
- Document Element
- Children
- Standalone
- ...

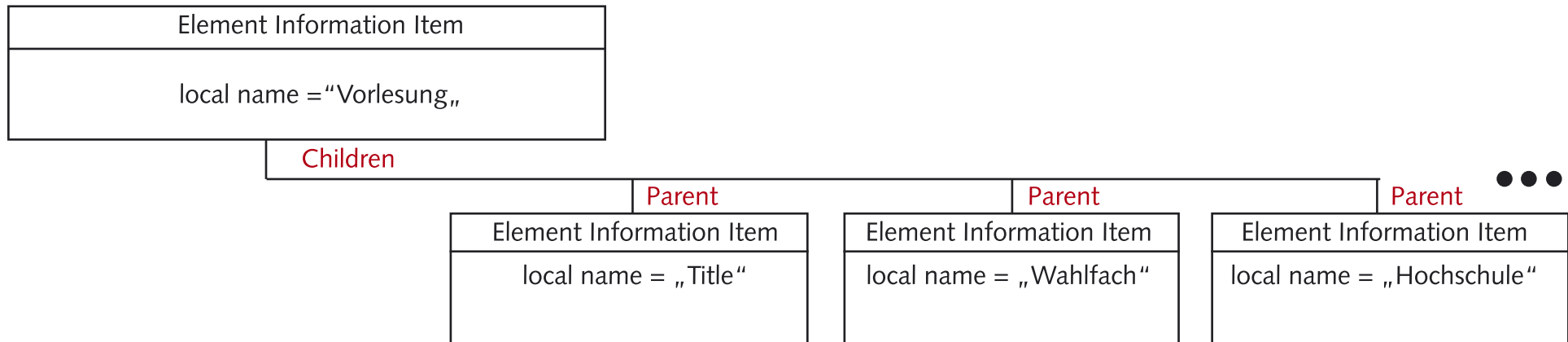


```
<?xml version="1.0" encoding="ISO-8859-15" standalone="yes"?>  
<Vorlesung>  
  <Wahlfach/>  
  <Titel beginn="2003-03-17T14:00:00+01:00">XML</Titel>  
  <Hochschule>Universität Bielefeld</Hochschule>  
  <Praktikum>Freiwilliger Übungsbetrieb</Praktikum>  
</Vorlesung>
```

## Element Information Item

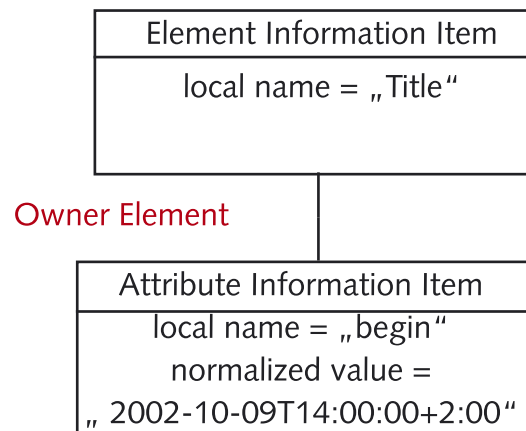
- Children
- Attribute
- Local name
- Namespace...

```
<?xml version="1.0" encoding="ISO-8859-15" standalone="yes"?>
<Vorlesung>
  <Wahlfach/>
  <Titel beginn="2003-03-17T14:00:00+01:00">XML</Titel>
  <Hochschule>Universität Bielefeld</Hochschule>
  <Praktikum>Freiwilliger Übungsbetrieb</Praktikum>
</Vorlesung>
```



## Attribute Information Item

- Owner
- NO Children
- Local name
- Namespace
- Normalized value
- ...



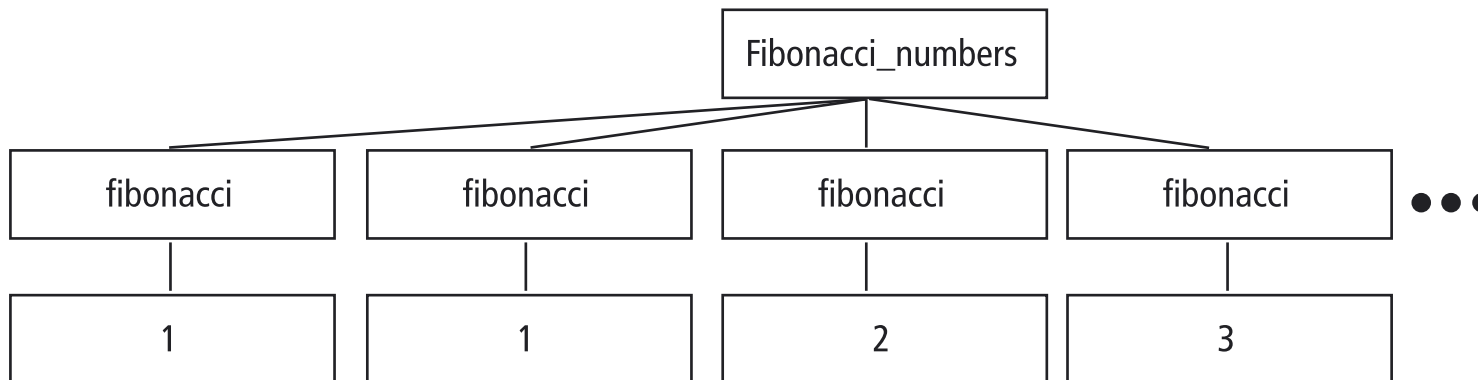
```
<?xml version="1.0" encoding="ISO-8859-15" standalone="yes"?>
<Vorlesung>
  <Wahlfach/>
  <Titel beginn="2003-03-17T14:00:00+01:00">XML</Titel>
  <Hochschule>Universität Bielefeld</Hochschule>
  <Praktikum>Freiwilliger Übungsbetrieb</Praktikum>
</Vorlesung>
```



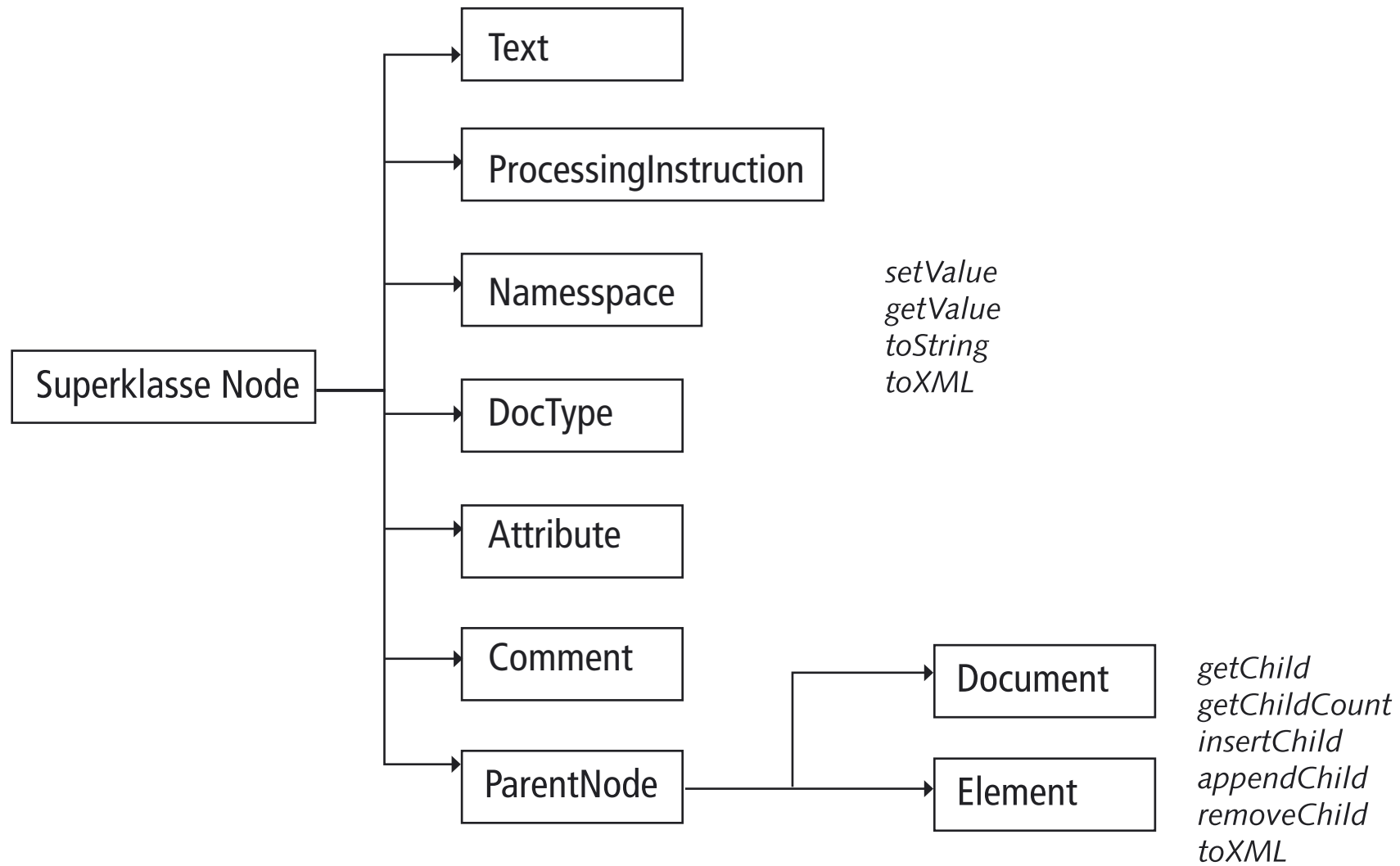
Umsetzung im DOM anhand von XOM

## Document Object Model

- programmiersprachenunabhängige Schnittstelle
- Baumstruktur
- Dokumentknoten stellt die gesamte Baumstruktur dar
- Elementknoten entspricht exakt einem Element in XML
- Attributknoten
- Textknoten



```
<?xml version="1.0"?>  
<Fibonacci_Numbers>  
  <fibonacci>1</fibonacci>  
  <fibonacci>1</fibonacci>  
  <fibonacci>2</fibonacci>  
  <fibonacci>3</fibonacci>  
  <fibonacci>5</fibonacci>  
</Fibonacci_Numbers>
```



```
...
import nu.xom.*;

public class FibonacciXML {

    ...
    ...

    Element root = new Element(„Fibonacci_Numbers“);

    for (int i = 1; i <= 10; i++) {
        Element fibonacci = new Element(„fibonacci“);
        fibonacci.appendChild(low.toString());
        root.appendChild(fibonacci);
    }
    Document doc = new Document(root);
    System.out.println(doc.toXML());
    ...
}
```

Ergebnis:

```
<?xml version="1.0"?>  
<Fibonacci_Numbers><fibonacci>1</fibonacci><fibonacci>1</fibonacci><fibonacci>2</fibonacci>...
```

## Code

```
...  
...  
    try {  
        Serializer serializer = new Serializer(System.out, „ISO-8859-1“);  
        serializer.setIndent(4);  
        serializer.setMaxLength(64);  
        serializer.write(doc);  
    }  
...  
...
```

## Ergebnis

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<Fibonacci_Numbers>  
    <fibonacci>1</fibonacci>  
    <fibonacci>1</fibonacci>  
    <fibonacci>2</fibonacci>  
    <fibonacci>3</fibonacci>  
    <fibonacci>5</fibonacci>  
    <fibonacci>8</fibonacci>  
    <fibonacci>13</fibonacci>  
    <fibonacci>21</fibonacci>  
    <fibonacci>34</fibonacci>  
    <fibonacci>55</fibonacci>  
</Fibonacci_Numbers>
```

## Code

```
...
Element root = new Element(„Fibonacci_Numbers“);
for (int i = 1; i <= 10; i++) {
    Element fibonacci = new Element(„fibonacci“);
    fibonacci.appendChild(low.toString());
    Attribute index = new Attribute(„index“, String.valueOf(i));
    fibonacci.addAttribute(index);
    root.appendChild(fibonacci);

    BigInteger temp = high;
    high = high.add(low);
    low = temp;
}
Document doc = new Document(root);
DocType doctype = new DocType(„Fibonacci_Numbers“, „fibonacci.dtd“);
doc.insertChild(doctype, 0);
....
```

## Ergebnis

```
<?xml version="1.0"?>
<Fibonacci_Numbers xmlns="">
  <fibonacci index="1">1</fibonacci>
  <fibonacci index="2">1</fibonacci>
  <fibonacci index="3">2</fibonacci>
  <fibonacci index="4">3</fibonacci>
  <fibonacci index="5">5</fibonacci>
  <fibonacci index="6">8</fibonacci>
  <fibonacci index="7">13</fibonacci>
  <fibonacci index="8">21</fibonacci>
  <fibonacci index="9">34</fibonacci>
  <fibonacci index="10">55</fibonacci>
</Fibonacci_Numbers>
```

```
....  
try {  
    Builder parser = new Builder();  
    Document doc = parser.build(args[0]);  
    System.out.println(doc.toXML());  
}  
catch (ParsingException ex) {  
    System.out.println(args[0] + „ is not well-formed.“);  
    System.out.println(ex.getMessage());  
}  
catch (IOException ex) {  
    System.out.println(  
        „Due to an IOException, the parser could not print „  
        + args[0]  
    ...  
}
```

## API-Design:

- Von Experten für Nichtexperten
- Es geht um XML, also soll es auch XML sein

## XOM- Design:

- Sicher, einfach und Wohlgeformtheit
- Aufbau in Klassen, keine Interfaces
- es gibt nur einen Weg

## Unterschied von XOM zu JDOM:

- XOM soll einfacher
- XOM soll absolut korrekt sein
- Node-Superclass
- JDOM Elemente enthalten eine Liste- XOM Elemente sind Bestandteil einer Liste

- XML ist Markup language
- Infoset ist semantische Basis von XML
- Infoset verändert Blickwinkel auf XML - sagt aus was wichtig ist und was nicht
- XML Infoset als Vermittler zwischen Api Designern und der XML-Spezifikation
- hilft APIs zu verstehen
- hilft XML zu nutzen
- hilft XML zu verstehen
- XOM nur eine weitere API?

Danke für die Aufmerksamkeit!

[cafeconleche.org/XOM/apidocs](http://cafeconleche.org/XOM/apidocs)

Aufgaben:

- parsen von hamlet.xml
- wieviele Akte hat das Stück?
- löschen des 1.Aktes

Jars + hamlet.xml:  
[4inloop.de/xom.tar](http://4inloop.de/xom.tar)